Non-Functional Requirements: Size Measurement and Testing with COSMIC-FFP

M. Kassab[1], O. Ormandjieva[2], M. Daneva[3], A. Abran[4]

[1,2]{moh_kass, ormandj}@cse.concordia.ca,
[3]m.daneva@utwente.nl, [4]Alain.Abran@etsmtl.ca

Abstract. The non-functional requirements (NFRs) of software systems are well known to add a degree of uncertainty to the process of estimating the cost of any project. This paper contributes to the achievement of more precise project size measurement through incorporating NFRs into the functional size quantification process. We report on an initial solution proposed to deal with the problem of quantitatively assessing the NFR modeling process early in the project, and of generating test cases for NFR verification purposes. The NFR framework has been chosen for the integration of NFRs into the requirements modeling process and for their quantitative assessment. Our proposal is based on the functional size measurement method, COSMIC-FFP, adopted in 2003 as the ISO/IEC 19761 standard. Also in this paper, we extend the use of COSMIC-FFP for NFR testing purposes. This is an essential step for improving NFR development and testing effort estimates, and consequently for managing the scope of NFRs. We discuss the merits of the proposed approach and the open questions related to its design.

1 Introduction

The increasing software complexity and competition that exist in the software industry have highlighted the need to consider non-functional requirements (NFRs) as an integral part of software modeling and development. According to IEEE software engineering standard 830-1998 [3], NFRs describe not what the software will do, but how it will provide the means to perform functional tasks; for example, software quality attributes software design constraints and software interface requirements. During requirements elicitation and analysis, NFRs tend to be stated in terms of either the qualities of the functional tasks or the constraints on them, which are expressed as functional requirements (FRs), as the former affect the semantics of the latter.

Empirical reports consistently indicate that improperly dealing with NFRs leads to project failures, or at least to considerable delays, and, consequently, to significant increases in the final cost [1, 2]. While estimating development effort is a major activity in managing the scope of the requirements, this activity has, by and large, been neglected for NFRs in practice. The need to deal comprehensively with the

effect of NFRs on the effort of building the software project generates the need to measure their functional size, as effort is a function of size [18]. In this paper, the use of the COSMIC-FFP [10, 11] functional size measurement method is proposed to quantify NFR size in a software project. To our knowledge, this is the first attempt to deploy such an approach for NFRs in a project and to generate test cases to verify them.

The NFR framework outlined in [4] has been chosen to illustrate the applicability of the proposed measurement method. It was the first framework to propose a process-oriented and qualitative decomposition approach for dealing with NFRs in requirements engineering (RE). A cornerstone of this framework is the concept of the "softgoal", which is used to represent the NFR. A softgoal is a goal that has no clear-cut definition or criteria to determine whether or not it has been satisfied. In fact, the framework speaks of softgoals being "satisficed" rather than satisfied, to underscore their ad hoc nature, with respect to both their definition and their satisfaction. One drawback of the softgoal approach implied in the NFR framework becomes apparent when we look at solution architecture tradeoffs. The term *softgoal* reflects the fact that extensive interdependencies exist between the various NFRs, and it is often not feasible to entirely fulfill each and every system goal. Tradeoffs must therefore be made. To understand these tradeoffs with respect to the NFR framework, the subgoals are decomposed into *operationalizations*, which provide both candidate design solutions for achieving the goal and the basis for weighing potential tradeoffs. However, the decision-making process for selecting from among different candidate operationalizations to satisfice a particular NFR is a qualitative process, which, typically, is not based on defined quantitative criteria. Furthermore, this process is only carried out informally, leaving undocumented the knowledge and the rationale that led to the decisions. This makes it difficult to trace back to the selection criteria on which those decisions were developed.

The above shortcomings underline the need to consider criteria which make it easier for analysts and software engineers to weigh the various design options and make tradeoffs in quantitative terms. In this paper, we address this need by suggesting that the softgoal concept in the context of the NFR framework be coupled with NFR functional size. Having the functional size stated for the operationalization softgoal in this way provides quantitative criteria for the decision-making task to select the most suitable operationalizations from the candidate alternatives. This paper also extends the use of COSMIC-FFP for NFR verification purposes by combining the functions measured by the COSMIC-FFP measurement procedure with a black box testing strategy. The test generation method from the earlier research is adopted for the purpose of generating scenario-based test cases from COSMIC-FFP models [28].

The rest of this paper is organized as follows: Section 2 presents related work and introduces the NFR framework. In section 3, the approach to measuring the size of NFRs is explained. Section 4 discusses the generation of test cases for NFR verification. Section 5 provides a critical discussion of the approach. Section 6 summarizes the key points of the solution proposal and discusses avenues for future research.

2 NFR Framework and Related Work

The NFR framework [4] is a process-oriented and goal-oriented approach aimed at making NFRs explicit and putting them at the forefront of the stakeholder's mind. Putting the framework in practice implies executing the following interleaved tasks, which are iterative:

1. Acquiring knowledge about the system's domain, FRs and the particular kinds of NFRs specific to that system of a particular system;
2. Identifying NFRs as NFR softgoals and decomposing them into a finer level;
3. Identifying the possible design alternatives for meeting NFRs in the target system as operationalizing softgoals;
4. Dealing with ambiguities, tradeoffs, priorities and interdependencies among NFRs and operationalizations;
5. Selecting operationalizations;
6. Supporting decisions with a design rationale;
7. Evaluating the impact of operationalization selection decisions on NFR satisfaction.

The operation of the framework can be visualized in terms of the incremental and interactive construction, elaboration, analysis and revision of a softgoal interdependency graph (SIG). Figure 1 presents an example of a SIG with NFR softgoals representing performance and security requirements for customer accounts in a credit card system.

In terms of related work, Paech et al. [6] recommend that FRs, NFRs and architecture be tightly co-developed and addressed in a coherent and integrated manner. These authors suggest that NFRs be decomposable into more refined NFRs and additional FRs, as well as architectural decisions. We adopt this proposal, while quantifying the NFRs as described in section 3.

3 Measuring the size of NFR

For the purposes of this research, we have chosen to use the functional size measurement method COSMIC-FFP [10] developed by the Common Software Measurement International Consortium (COSMIC) and now adopted as an international standard (ISO/IEC 19761 [11]). Our solution proposal is presented in Figure 2. It shows how the functional size measurement of NFRs is integrated into the NFR framework. We see the NFR framework as the vehicle for eliciting, documenting and operationalizing NFRs. We then propose that COSMIC-FFP be applied to obtain the NFR functional size data. These data are then provided to the relevant stakeholders to assist them in their decision-making process.

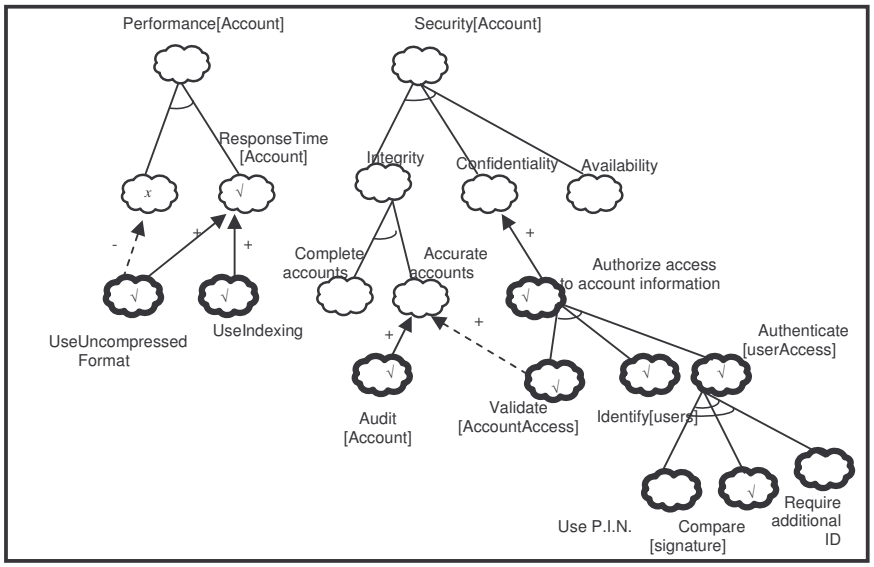

Fig. 1. Softgoal interdependency graph for performance and security in a credit card system [4]

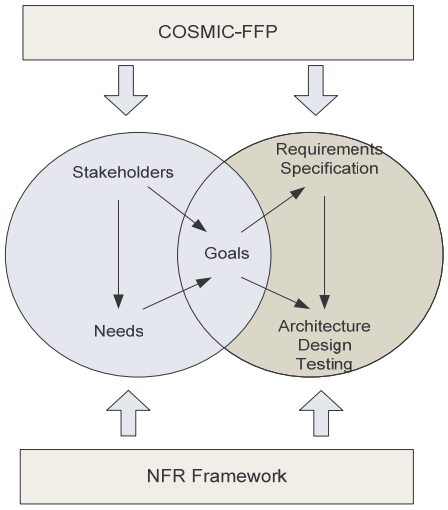

Fig. 2. The solution proposal: a high-level view.

3.1 The COSMIC_FFP method

The COSMIC-FFP measurement method conforms to all ISO requirements (ISO 14143-1 [12]) for functional size measurement, and addresses some of the major theoretical weaknesses of the earlier Function Point Analysis techniques like Albrecht's Function Points [13], which dates back almost 30 years to a time when software projects were much smaller and less complex. COSMIC-FFP, in contrast to [13], focuses on the "user view" of functional requirements, and is applicable throughout the development life cycle, from the requirements phase right through to the implementation and maintenance phases.

The process of measuring software functional size using the COSMIC-FFP method implies that the software functional processes and their triggering events be identified. In COSMIC-FFP, the unit of measurement is the data movement, which is a base functional component that moves one or more data attributes belonging to a single data group. It is denoted by the symbol *Cfsu* (Cosmic Functional Size Unit). Data movements can be of four types: Entry, Exit, Read or Write. The functional process is an elementary component of a set of user requirements triggered by one or more triggering events, either directly or indirectly, via an actor. The triggering event is an event occurring outside the boundary of the measured software and initiates one or more functional processes. The subprocesses of each functional process are sequences of events; a functional process comprises at least two data movement types: an Entry plus at least either an Exit or a Write. An Entry moves a data group, which is a set of data attributes, from a user across the boundary into the functional process, while an Exit moves a data group from a functional process across the boundary to the user requiring it. A Write moves a data group lying inside the functional process to persistent storage, and a Read moves a data group from persistent storage to the functional process. Figure 3 illustrates the generic flow of data attributes through software from a functional perspective.

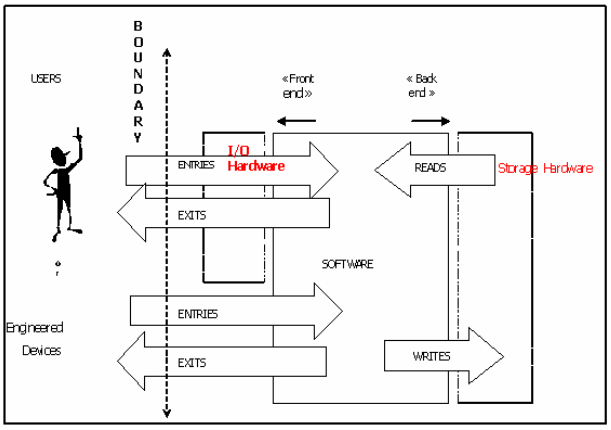

Fig. 3. Generic flow of data attributes through software from a functional perspective [10]

3.2 Size measurement on NFRs

In our approach, we apply COSMIC-FFP to NFRs stated in verifiable terms. This means that NFRs are stated in terms of crisp indicators with defined acceptable values; thus, it is possible to verify the satisfaction level of those NFRs by comparing the acceptable values with the actual achieved values.

Two views on the size of NFRs are addressed: (1) First, COSMIC-FFP is used to measure the functional size for those operationalizations that correspond to functional processes/functions; and (2) then COSMIC-FFP is used to measure the functional size of the quality control that NFRs require at runtime. NFR quality control is the operation/function that aims to verify the satisfaction of NFRs at runtime (e.g. comparing the acceptable with the actual values).

We state that the size of verifiable NFRs is the sum of both views explained above. The addition of the size values is theoretically valid because COSMIC-FFP size has a unique unit of measurement, the Cfsu, thus the COSMIC-FFP size measure is at least on the ratio scale. For further discussion on the scale types and the representational theory of measurement, see [27].

Illustration. The NFR size measurement approach is illustrated on the "availability" NFRs from the credit card system accounts example (see Figure 1).

Two functional processes have been identified in the COSMIC-FFP model for NFR availability, one for each view of the NFR size measurement explained above. The functional processes are: i) availability quantification operationalization, for sizing the reliability of the availability; and ii) availability monitoring, for sizing its monitoring at runtime. Our assumption here is that the availability measurement model is time-dependent, and thus requires a record of the history of system failures, which is stored by the Credit Card System. Moreover, each change in the failure history triggers an update of the availability level, which is modeled as the Availability Quantification process in Table 1. At the same time, the Credit Card System can request a report on the current availability level. This request triggers the Availability Monitoring process (see Table 1), which analyzes the current availability level and issues an "acceptable level" or "critical level" report, depending on the analysis results.

The COSMIC-FFP functional processes and their functional size calculation are illustrated in Table 1. (In this table, the heading of the fifth column, DTM, stands for Data Movement Type.) The total COSMIC-FFP functional size of the availability operationalization and monitoring is 7 Cfsu.

Similarly, we perform this measurement for all non-decomposable operationalizations that correspond to functional operations/functions. Calculating the functional size of the NFRs is a bottom-up measuring process, in which the selected operationalizations are aggregated to calculate the sub-NFRs and the respective NFRs until the final value is obtained. The functional size of the control functions is added during this process when applicable (see Figure 4 for an illustration of the process).

This task should be performed immediately following task 3 and prior to task 4 in the NFR framework process presented in section 2. The measurement data will provide the rationale required for selecting the appropriate operationalizations. For example, the two operationalizations "Compare Signature" and "Use P.I.N." are 3 Cfsu and 2 Cfsu in size respectively; we have to choose one operationalization to

satisfy “Authenticate”, and then we may consider choosing “Use P.I.N.”, as it has a smaller functional size and will thus require less effort/cost to be implemented.

Table 1. COSMIC-FFP Data Movements

Process ID	Process description	Triggering event	Data movement identification	Data Group	DMT	Cfsu
1.1	Availability Quantification	New Failure Data Signal	Receive triggering event	New Failure Data Signal	E	1
			Read Failure History	Failure History	R	1
			Write Current Availability Level	Current Availability	W	1
Functional size in Cfsu				**=**	**ΣCfsu**	**3**
1.2	Availability Monitoring	Monitor Availability Signal	Receive triggering event	Monitor Availability Signal	E	1
			Read Target Availability Level	Target Availability	R	1
			Read Current Availability Level	Current Availability	R	1
			Send acceptable/critical level message	Report	X	1
Functional size in Cfsu				**=**	**ΣCfsu**	**4**
Total COSMIC-FFP points in Cfsu				**=**	**ΣCfsu**	**7**

At the same time, some of the NFRs, such as availability, require continuous control (quantification and analysis of the measurement data) at runtime to obtain feedback on the overall quality of the application. We consider monitoring the quality NFR as an NFR subprocess attached to the corresponding softgoal, and introduce a special symbol to denote such a subprocess: 

The subprocesses are further refined into Quantification and Monitoring operationalizations, the size of which is measured with the COSMIC-FFP method in case these subprocesses correspond to functions (see, for instance, Table 1). This approach is illustrated on the Availability NFR (see Figures 4 and 5).

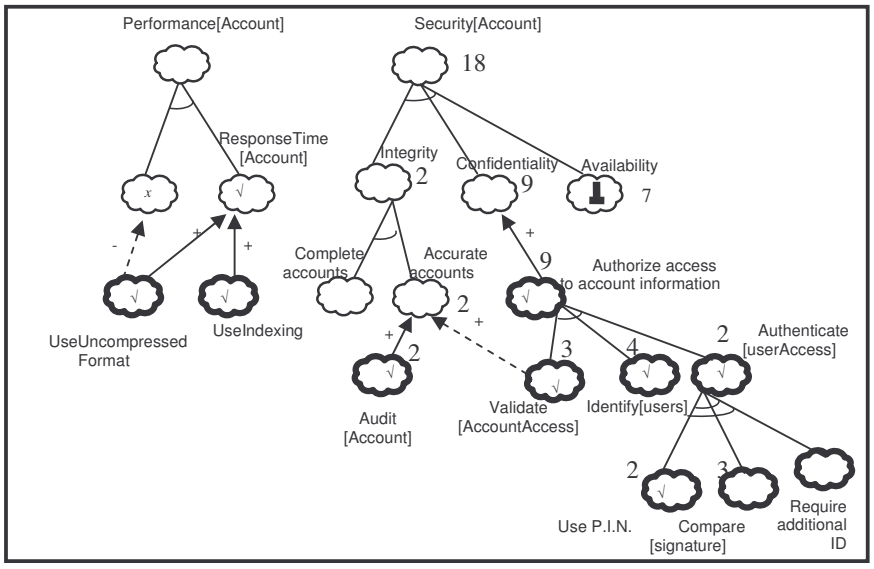

Fig. 4. Calculating functional size for NFRs

Black-box scenario-based test cases for NFR are then derived from the corresponding COSMIC-FFP models, which helps ensure conformance of the final product to user expectations. The test cases are generated through a mapping of scenarios to sequences of events in time (or data movements in COSMIC-FFP), as described below.

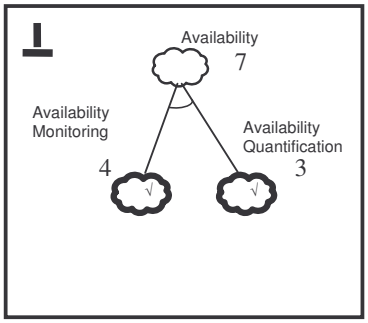

Fig. 5. Refinement of the Availability control subprocess

4 Testing NFRs

Scenario-based testing is a typical black-box testing methodology at the system level [28]. One of its greatest benefits is that it provides testers with a set of assets that can directly drive the testing process. The scenario-based black-box test cases for verifying NFRs are generated through mapping the corresponding COSMIC-FFP functional processes to sequences of events in time (or data movements in COSMIC-FFP). The procedure for generating a test case is illustrated on availability monitoring (see section 3.2), which in this instance is generated from the corresponding functional processes (see Table 1, process 1.2), and in turn mapped to the following sequence of events:

Test case ID	Test case description
t1	Receive Monitor Availability Signal, Read Target Level, Read Current Level, Send Result

Next, the specific conditions that would cause the test case to execute are identified, and real data values are supplied. The details of the test derivation from COSMIC-FFP models are described in [28].

5 Discussion

While our proposed solution to measuring the size of NFR makes sense and sounds intuitive, it is far from being issue-free or straightforward to apply. If its purpose is to provide estimators with more realistic size and effort estimates, then we have to make a fine distinction between the two directions estimators may take in quantifying NFRs.

Some publications [22, 23] suggest that, in order for estimators to be able to obtain size and effort numbers for the NFRs in a project, the NFRs must be first decomposed into a series of corresponding FRs. Once this has been done, a functional size measurement method is considered to be the suitable vehicle for quantifying the contribution of NFRs to software size, and, ultimately, to the effort it would take to build the software project. In these publications, it is assumed that it makes sense to decompose all NFRs into FRs. Recently, however, this assumption has become a subject of discussion among some RE researchers [14, 15, 16], who support the position that not all NFRs should be decomposed into FRs. Clearly, there is agreement in the literature that the majority of NFRs can and should be decomposed into FRs, but these RE researchers maintain that there are specific types of NFRs which cannot be decomposed into FRs. Specifically, the goal-oriented RE community [14,15,16] considers that NFRs should not be decomposed into FRs if: (i) the NFRs are normative, that is, if they stipulate how the actor in the system environment shall behave when interacting with the system [16]; or (ii) the NFRs serve as criteria for making architectural design choices; that is, the function of these NFRs is to help evaluate alternatives. Examples of such requirements are the statements "*Time zone information shall be kept together with a local timestamp*" [17] and "*For all Quebec*

users, the system's language shall be French". In a typical requirements document, these NFRs would be stated in textual form and would not be present, for example, in a requirements diagram (e.g. a use case) which documents the business process and data flows that the system under development must support. Certainly, we can decompose the NFR from "*The system's language shall be French*" to an FR like "*Each Quebec user is offered the functionality to select a language*," "*…to select all documents that should use this language*," "*…to generate reports in this language*," and so on.

However, it may well make more sense to consider the NFR as a criterion for exploration and for making choices among alternative architecture options. This consideration is motivated by the following observations: (1) the above decomposition into an FR (which is needed for effort estimators and is used for determining size) refers to functionality that the user did not ask for at the time of RE; (2) the NFR is a norm to which the user and the system must conform [16], in a bilingual environment (such as Canada or Belgium), for example, where the choice of language is not dictated by user request but by corporate standards and national regulations; and (3) the language of an application tells us about the project context, hence it may point to a contextual factor that may well be a source of risk [18] in terms of obtaining realistic size and effort estimates. Moreover, it should be possible for global applications, like ERP-packaged solutions, which typically produce language-specific reports for specific user groups, to prepare reports in the language specified by the user group. The design architects must then choose a way to set up such a multi-language NFR.

Drawing on this analysis of the RE literature, we incorporated John Mylopolous' view of NFRs as architectural design selection criteria [14] in our approach to estimating the size and effort associated with NFRs. Our position is also based on the recommendations of software measurement practitioners [18], who maintain that we, the estimators, need to know the project context first and then use our knowledge of that context to arrive at better estimates. It is our understanding, and our position in this paper, that, if the knowledge of the context of how a system will be used is reflected, and captured in the NFRs, then those NFRs that are not decomposable into FRs should be used as criteria for making design decisions. These decisions are made at two levels [18]: at the micro level (for example, how to design a particular module of a system), and at the macro level (for example, which software architecture to employ). Our position also implies that, whenever we make an architectural design decision, we can potentially affect the accuracy of the cost estimates, since making those decisions introduces uncertainty into the estimation process [18]. This issue is aggravated in the RE phase, where we typically have an incomplete picture of the project context. As a result, there is much more uncertainty surrounding the effort required to satisfactorily develop the capabilities to which the NFRs refer. Because we have to judge how significant these uncertainties (due to NFRs) are, we have to account for them in reporting the final project size assessment.

Therefore, we take into account that cost estimation needs are expected to vary based on the nature of the project, a fact which will also be reflected in the NFRs. For example, a brand-new technology, like implementing a cross-business unit-integrated Enterprise Resource Planning (ERP) system [19], the users of which do not know what their NFRs look like, has more sources of uncertainty than an ERP upgrade

project. We can reduce these uncertainties significantly by having the NFR measurement activity explicitly included in reporting the total project size measurement value made up of both FR and NFR size. Consequently, we will establish a more precise estimation of the actual development effort of the system. We assume that, based on particular estimation needs, we may even consider using different sizing methods (COSMIC or FP and their variants) to address the NFR issue in a specific project.

6 Summary and future research plans

This paper reports on an initial solution to deal with the problem of quantitatively assessing the NFR modeling process early in the project, and of generating test cases for NFR verification purposes. The NFR Framework approach has been chosen to illustrate the integration of NFRs into the requirements modeling process, and for describing NFR quantitative assessment. Our proposal relies on the COSMIC-FFP functional size measurement method.

To the best of our knowledge, the software industry lacks quantitative effort estimation methods for NFRs, and would certainly benefit from the precise and objective size measurement approach proposed in this paper. This is the motivation for three research activities planned for the near future:

- Determine how the size of NFRs impacts the total project cost,
- Conduct case studies to assess the usefulness of the technique (for example, to research what happens when models of operationalized NFRs become larger,
- Derive guidelines for how to systematically deal with those NFRs that can be decomposed into FRs up to a certain level.

References

1. Lindstorm, D.R.: Five Ways to Destroy a Development Project. IEEE Software, (1993) 55-58.
2. Breitman, K. K, Leite J.C.S.P. and Finkelstein, A.: The World's Stage: A Survey on Requirements Engineering Using a Real-Life Case Study. Journal of the Brazilian Computer Society No 1 Vol. (1999) 13-37.
3. IEEE Std. 830-1998: IEEE recommended practice for software requirements specifications. IEEE Transactions on Software Engineering, (1998).
4. Chung, L., Nixon, B. A., Yu, E. and Mylopoulos J.: Nonfunctional Requirements in Software Engineering, Kluwer Academic Publishing, (2000).
5. Andrew, J: An Approach to Quantitative Non-Functional Requirements in Software Development, Proceedings of the 34th Annual Government Electronics and Information Association Conference, (2000).
6. Paech, B., Dutoit, A., Kerkow, D. and von Knethen, A.: Functional requirements, non-functional requirements and architecture specification cannot be separated -- A position paper, REFSQ (2002).

7. Moreira, A., Araujo, J. and Brito, I.: Crosscutting Quality Attributes for Requirements Engineering, In 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, (2002) 167–174.
8. Park, D. and Kand, S.: Design Phase Analysis of Software Performance Using Aspect-Oriented Programming, In 5th Aspect-Oriented Modeling Workshop in Conjunction with UML 2004, Lisbon, Portugal, (2004).
9. Adelman L. and Donnell M.L.: Evaluating decision support systems: A General framework and case study, In S.J. Andriole (Ed.), Microcomputer Decision Support Systems: Design, Implementation, and Evaluation. Wellesley, MA: QED Information Science, (1986) 285-310.
10. Abran A., Desharnais, J.-M. , Oligny, S., St-Pierre, D. and Symons, C. : COSMIC FFP – Measurement Manual (COSMIC implementation guide to ISO/IEC 19761:2003), École de technologie supérieure – Université du Québec, Montréal, Canada, (2003), URL: http://www.gelog.etsmtl.ca/cosmic-ffp/manual.jsp .
11. ISO/IEC 19761. Software Engineering : COSMIC-FFP - A functional size measurement method, International Organization for Standardization – ISO, Geneva, (2003).
12. ISO 14143-1. : Functional size measurement - Definitions of concepts, International Organization for Standardization – ISO, Geneva, (1988).
13. Albrecht, A.J. and Gaffney, J.E.: Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, IEEE Trans. Software Eng. vol. SE-9, no. 6, Nov. (1983), 639-648
14. Mylopoulos, J.: Goal-oriented Requirements Engineering, Keynote speech at the 14th IEEE International Conference on Requirements Engineering, IEEE Computer Society Press, (2006).
15. Glinz, M.: Rethinking the Notion of Non-Functional Requirements, Proc. of the 3rd World Congress for Software Quality, Munich, Germany, (2005).
16. Wieringa, R.: The Declarative Problem Frame: Designing Systems that Create and Use Norms, Proc. of the 10th IEEE International Workshop on Software Specification and Design, IEEE Computer Society Press, (2000), 75-85.
17. Wroblewski, M.: Quality Governance and Production, Software Quality and Service-oriented Architecture, Proc of 9th International Conference on Quality Engineering in Software Technology, Berlin, (2006), 333-344.
18. Pfleeger, S. L., F. Wu and R. Lewis: Software Cost Estimation and Sizing Methods: Issues and Guidelines, RAND Corporation, (2005).
19. Daneva M.: ERP Requirements Engineering Practice: Lessons Learnt, IEEE Software, 21(2), 26-33.
20. Mylopoulos, J., Chung, L. and Nixon, B.: Representing and Using Nonfunctional Requirements: A process Oriented Approach, IEEE Trans. S.E. 18, (1992) 483-497.
21. Rosa, N.S., Cunha, P.R.F., and Justo: G.R.R.: ProcessNFL: A language for Describing Non-Functional Properties, Proc. 35th HICSS, IEEE Press (2002).
22. ISBSG, Practical Software Estimation, 2nd Edition: International Software Benchmarking Standard Group, (2006).
23. FISMA, Experience Situation Analysis, Finnish Software Metrics Association, (2001), http://www.fisma.fi/wp-content/uploads/2006/09/fisma_situation_analysis_method_nd21.pdf
24. Alves, C., Franch , X., Carvallo, J.P. and Finkelstein, A.: Using Goals and Quality Models to Support the Matching Analysis During COTS Selection, Proc. of the IEEE Int. Conf. on Component-based Systems (2005), 146-156

25. Jureta, I., Faulkner, S., Schobbens, P.-Y: A More Expressive Softgoal Conceptualization for Quality Requirements Analysis, Proc. of IEEE Int. Conf. on Conceptual Modelling (RE06), 281-295.
26. Kaiya, H., Osada, A., Kayjiri, K.: Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems, Proc. of the IEEE Int. Conf. on Requirements Engineering (RE04), 112-121.
27. Norman, E., Fenton, Shari Lawrence Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, PWS Publishing, 2nd edition, revised printing, (1998), ISBN 0-534-95425-1.
28. Abu Talib, M., Ormandjieva, O., Abran, A., Khelifi, A., Buglione, L.: Scenario-based Black-Box Testing in COSMIC-FFP: a Case Study, ASQ Software Quality Professional Journal 8 (3), (2006) 23-33.